

Objetivos

- Aprender cómo leer y procesar archivos CSV en Python
- Hacer recorridos parciales y totales sobre listas
- Usar listas de diccionarios

Preparación

1. Descargue los siguientes archivos de la actividad de Brightspace, y guárdelos en una misma carpeta en su computador:

- **pokemon_funciones.py**: Módulo de python sobre el que vamos a trabajar
- **pokemon_consola.py**: Script que ejecuta las funciones del primer archivo
- **pokemon.csv**: Archivo CSV con la información de los pokemones
- **pokemon_estadisticas.csv**: Archivo CSV con las estadísticas de ataque y defensa de cada pokemon.

Primera parte: Cargando archivos

2. Abra **pokemon_funciones.py** en Spyder

Vamos a aprender primero a cargar un archivo desde python. La sintaxis para hacerlo es sencilla, sólo tenemos que usar una variable y llamar a la función **open** de python.

Función cargar_pokemones()

```
def cargar_pokemones()->list:
    lista_pokemon = []
    # TODO: cargar archivo de pokemones

    # TODO: leer el encabezado y guardarlo en una variable llamada atributos
    # TODO: crear los pokemones y meterlos en la lista
    # TODO: cerrar el archivo

    return lista_pokemon
```

3. Agregue la siguiente línea después del **TODO** respectivo:

```
# TODO: cargar archivo de pokemones
archivo_pokemones = open('pokemon.csv')
```

Ahora vamos a leer la primera línea del archivo, que tiene los nombres de las columnas.

4. Agregue la siguiente línea a la función, después del **TODO**:

```
# TODO: leer el encabezado y guardarlo en una variable llamada atributos
primera_linea = archivo_pokemones.readline()
```

Si abrimos el archivo **pokemon.csv**, vemos que esta es la primera línea del archivo CSV:

id,identifier,species_id,height,weight,base_experience,order,is_default

Esta cadena la guardamos en la variable **primera_linea**.

5. Agregue la siguiente línea a su código, después de la línea del punto 4:

```
atributos = primera_linea.replace("\n", "").split(",")
```

Aquí lo que hicimos fue convertir esa cadena en una lista de los valores separados por comas:

```
['id', 'identifier', 'species_id', 'height', 'weight', 'base_experience', 'order', 'is_default']
```

Note que usamos el método **replace**, ya que el string de cada línea tiene al final un salto de línea. Así sería nuestra lista si no hacemos el **replace**:

```
['id', 'identifier', 'species_id', 'height', 'weight', 'base_experience', 'order', 'is_default\n']
```

6. Ahora vamos a procesar desde la segunda línea del archivo. Inserte la siguiente línea de código después del tercer **TODO**:

```
# TODO: crear los pokemones y meterlos en la lista
linea = archivo_pokemones.readline()
```

Este código lee la segunda línea, después del encabezado.

7. Ahora inserte el siguiente ciclo inmediatamente después:

```
while len(linea) > 0:
    datos = linea.replace("\n", "").split(",")
    pokemon = {}
```

Aquí estamos cogiendo la línea y eliminando los saltos, y separando por comas. Luego, estamos creando un diccionario vacío.

8. Después de crear el diccionario inserte el siguiente ciclo:

```
for i, a in enumerate(atributos):
    pokemon[a] = datos[i]
```

A pesar que sólo son dos líneas, aquí estamos haciendo varias cosas. Por ejemplo, note que estamos llamando a la función **enumerate**, y le damos como argumento la lista **atributos** del punto 5. Esta función es parecida a **range**, pero me brinda tanto un índice (en este caso **i**), así como el elemento actual de la lista (en este caso **a**).

Dentro del ciclo estamos asignando al diccionario **pokemon** cada uno de los atributos que hacen parte de la línea. Usamos como llave el nombre del atributo, y como valor accedemos a la posición **i** de la lista llamada **datos**. Esto es posible porque ambas listas tienen el mismo tamaño; por ejemplo, para el primer pokemon, el cuarto elemento en la lista **atributos** es "height", y el cuarto elemento de la lista **datos** es 7:

ejemplo

9. Finalmente, agregue las siguientes líneas de código afuera del ciclo **for** creado anteriormente:

```
lista_pokemon.append(pokemon)
linea = archivo_pokemones.readline()
```

Aquí estamos agregando el nuevo diccionario a la lista completa de pokemones. Luego, estamos leyendo **la siguiente línea** del archivo CSV. Como esto está dentro del **while**, todas las instrucciones se repiten mientras hayan líneas en el archivo.

Todo lo que agregamos debería quedar indentado de la siguiente manera:

```
# TODO: crear los pokemones y meterlos en la lista
linea = archivo_pokemones.readline()
while len(linea) > 0:
    datos = linea.replace("\n", "").split(",")
    pokemon = {}
    for i, a in enumerate(atributos):
        pokemon[a] = datos[i]
    lista_pokemon.append(pokemon)
    linea = archivo_pokemones.readline()
```

Agregue el siguiente código después del último **TODO** de la función:

```
# TODO: cerrar el archivo
archivo_pokemones.close()
```

Siempre que lea archivos usando la función **open** es importante que cierre la conexión con el archivo, usando el método **close**.

Desarrollo

10. Ya con lo anterior tenemos cargada la lista de pokemones. Vaya al archivo **pokemon_consola.py** y ejecútelo. Le debe salir los siguientes pokemon:

Nombres de los primeros 20 pokemon: *bulbasaur, ivysaur, venusaur, charmander, charmeleon, charizard, squirtle, wartortle, blastoise, caterpie, metapod, butterfree, weedle, kakuna, beedrill, pidgey, pidgeotto, pidgeot, rattata, raticate*


Función buscar_pokemon()

11. Complete la función **buscar_pokemon** para que me retorne el diccionario del pokemon dada una lista y un identificador. Para ello vamos a hacer un **recorrido parcial** sobre la lista de pokemones que entra como parámetro, usando como **centinela** la variable **buscado**. El código debe ir entre el comentario del **TODO** y el **return**:

```
def buscar_pokemon(lista: list, id: str) -> dict:
    i = 0
    buscado = None
    # TODO: Hacer un recorrido parcial sobre la lista
    return buscado
```

Función cargar_estadisticas()

Ahora vamos a leer desde Python otro archivo que contiene las estadísticas de cada pokemon. Este archivo se llama **pokemon_estadisticas.csv**:

ejemplo

12. Vaya a la función **cargar_estadisticas** y examine el código. Vamos por partes:

```
def cargar_estadisticas(lista_pokemon:list)->list:
    nombres_estadisticas = {
        "1": "hp",
        "2": "attack",
        "3": "defense",
        "4": "special-attack",
        "5": "special-defense",
        "6": "speed",
        "7": "accuracy",
        "8": "evasion"
    }
```

nombres_estadisticas es un diccionario con los identificadores de las estadísticas y sus nombres respectivos. Este diccionario lo vamos a usar para pasar de identificador a llave para la estadística del pokemon. Esto lo tenemos que hacer porque las estadísticas de cada pokemon están en varias líneas del archivo CSV.

13. Sigamos examinando función **cargar_estadisticas**:

```
archivo_estadisticas=open('pokemon_estadisticas.csv')
linea = archivo_estadisticas.readline()
linea = archivo_estadisticas.readline()
```

Primero abrimos el archivo de la misma forma que el otro. Sin embargo, aquí no nos interesa el encabezado de la tabla: aquí esquivamos esa línea y continuamos desde la segunda. Por eso llamamos dos veces al método **readline()**.

14. Complete el código después del **TODO** dentro del **while**:

```
while len(linea) > 0:
    # TODO: añadir las estadísticas al pokemon

    linea = archivo_estadisticas.readline()

    archivo_estadisticas.close()
```

Píense qué tenemos que hacer, inspirado en lo que aprendimos con la función **cargar_pokemones**. Abra el archivo **pokemon_estadisticas.csv**. Nos interesan los tres primeros valores de cada línea: el ID del pokemon, el ID de la estadística, y el valor correspondiente.

Segunda parte: Capturando pokemones

Ya tenemos todos los datos de los pokemones en la lista. Este es un ejemplo de un pokemon aleatorio de la lista:

```
{'id': '10070',
'identifier': 'sharpedo-mega',
'species_id': '319',
'height': '25',
'weight': '1303',
'base_experience': '196',
'order': '412',
'is_default': '0',
'hp': '70',
'attack': '140',
'defense': '70',
'special-attack': '110',
'special-defense': '65',
'speed': '105'}
```

Función capturar_10_pokemones()

15. Complete la función **capturar_10_pokemones**, que me retorne una lista de 10 pokemones escogidos aleatoriamente de la lista de todos los pokemones.

```
def capturar_10_pokemones(lista:list):
    capturados = []
    # Acceder a 10 elementos aleatorios usando for in range(0,10)
    return capturados
```

Este es un ejemplo que nos imprimiría los identificadores y nombres de la lista que nos retorna la función:

```
resultado = capturar_10_pokemones(lista_pokemon)
for p in resultado:
    print(p['id'],p['identifier'])
```

```
664 scatterbug
5 charmeleon
614 beartic
777 togedemaru
239 elekid
237 hitmontop
10030 gourgeist-small
384 rayquaza
104 cubone
180 flaaflly
```

Tercera parte: Armando el equipo

Ahora nos interesa tener un equipo variado de pokemones para nuestras peleas. Vamos a hacer tres funciones que nos ayudarán a lograr esto.

16. Realice una función que retorne el pokemon que tiene el mayor valor para cierta característica. Si existen varios con igual valor máximo, retorne el primero.

```
def dar_maximo_pokemon(lista_pokemon:list,caracteristica:str)->dict:
    # TODO dada una caracteristica, buscar en la lista el pokemon que tiene el mayor valor

    return None
```

Por ejemplo, este sería el pokemon con más ataque:

```
dar_maximo_pokemon(lista_pokemon,'attack')
```

```
{'id': '10043',
'identifier': 'mewtwo-mega-x',
'species_id': '150',
'height': '23',
'weight': '1270',
'base_experience': '351',
'order': '231',
'is_default': '0',
'hp': '100',
'attack': '190',
'defense': '100',
'special-attack': '154',
'special-defense': '100',
'speed': '130'}
```

17. Realice una función que retorne el pokemon que tiene el menor valor para cierta característica. Si existen varios con igual valor mínimo, retorne el primero.

```
def dar_minimo_pokemon(lista_pokemon:list,caracteristica:str)->dict:
    # TODO dada una caracteristica, buscar en la lista el pokemon que tiene el menor valor

    return None
```

Por ejemplo, este sería el pokemon con menos ataque especial:

```
dar_minimo_pokemon(lista_pokemon,'special-attack')
```

```
{'id': '213',
'identifier': 'shuckle',
'species_id': '213',
'height': '6',
'weight': '205',
'base_experience': '177',
'order': '294',
'is_default': '1',
'hp': '20',
'attack': '10',
'defense': '230',
'special-attack': '10',
'special-defense': '230',
'speed': '5'}
```

18. Realice la función **hacer_equipo_balanceado**, que me retorne una tabla con los siguientes pokemones:

- El pokemon con más ataque (**attack**)
- El pokemon con más defensa especial (**special-defense**)
- El pokemon más **débil** (**hp**)
- El pokemon más **lento** (**speed**)
- El pokemon más **alto** (**height**)
- El pokemon más **pequeño** (**height**)
- El pokemon más **pesado** (**weight**)

```
def hacer_equipo_balanceado(lista_pokemon:list)->str:
    # TODO Crear un equipo llamando a los metodos creados anteriormente,
    # cumpliendo los requerimientos del enunciado

    return ''
```

Este sería el resultado esperado:

	nombre	HP	ataque	defensa	defensa-esp	velocidad	altura	peso
mewtwo-mega-x	106	190	100	100	130	23	1270	
shuckle	20	10	230	230	5	6	205	
shedinja	1	90	45	30	40	8	12	
shuckle	20	10	230	230	5	6	205	
wailord	170	90	45	45	60	145	3980	
joltik	50	47	50	50	65	1	6	
cossmoem	43	29	131	131	37	1	9999	